Centralized Control of a Heterogeneous and Decentralized Computing Landscape
(DRAFT)

Brett Viren[*], Richard Casella, Todd Corsa, James Fung, Lisa Soto

March 15, 2006

Abstract

BNL has a large variety of decentralized and network accessible devices. For these devices to be secure and to fulfill DOE requirements they must be registered, checked for potential local and network based exploits and be assessed that they conform to baseline configurations. A system has been developed that leverages the centrally controlled network infrastructure to enforce these goals while still respecting the required diversity of the computing landscape. This note describes the elements of the BNL Centralized Computing Control (C3) system.

[*]Contact: bv@bnl.gov

Contents

1 Introduction

Brookhaven National Lab provides facilities for a wide variety of scientific research and development performed in many fields of study. To support this variety, a heterogeneous and decentralized computing landscape is a must. But, to assure all devices on the network are known and meet minimum security baselines requires centralized information and control. This note describes the Centralized Computing Control (C3) system developed and in use at BNL which provides these assurances.

1.1 Design Guidelines

There are some basic directives guiding the design of the BNL C3 system.

1. Know what is on the network.
2. Test for known network based vulnerabilities.
3. Test for known host based vulnerabilities and assess that systems implement baseline configuration requirements.
4. Record all the above in a central database.
5. Deny network access to unknown or non compliant systems.

Centralized control is assured by the last item. Full network access is denied unless the system is registered and has had network based and host based assessment scans performed. Denial of service is implemented by first controlling DNS and/or DHCP on an IP number basis. This will escalate to disabling the wall jack servicing the errant system.

Continued access is dependent on ongoing scans being performed and findings remediated. The network scan uses Nessus[1] a web front end developed at BNL and database storage. For unix-like systems, the host based scanner, "Ordo" has been developed in-house. For MS Windows, SMS is used. The following sections detail each of these major elements.

2 Back-end Database

2.1 Overview

The main database back-end stores device registration information, scan results and security status. This data is used to locate the device and its responsible individuals, remediate scan findings, and allow or deny network access.

2.2 Definition Of A "System"

The database back-end is centered around an abstract `DEVICE_ID` which is meant to correspond as closely as possible to a device's physical "box". A device may have a number of network interface cards (NICs) each of which presents a unique MAC address or may gain new NICs due to hardware changes. A device can also run multiple instances of operating system, either concurrently or sequentially. Finally, a device may change its IP address if it uses DHCP or if it has been re-purposed.

While this abstract `DEVICE_ID` supports such variety, due to how data is collected, it is not easy to always guarantee a one to one correspondence between `DEVICE_ID` and "box". In general this must be done "by hand" relying on the diligence of the system administrator. For systems running unix-like OS, this correlation is done automatically through the Ordo host based scanner (section 5).

[1] http://www.nessus.org/

2.3 The Tables

The tables fall into a few categories relating to how their data is collected and used.

- System registration tables.
- Network based scan tables. These hold results and remediation state of the Nessus scans.
- Host based scan tables. These hold results, remediation state and support data for the Ordo scans. In particular, they enumerate all NICs and their MAC/IP addresses in a device.
- Last seen on network time stamps (ARP data). These tables match a MAC address to its most recently used IP address.
- Firewall conduit tables. These hold information (CSMIS) on what devices are authorized to have ports visible through firewalls.

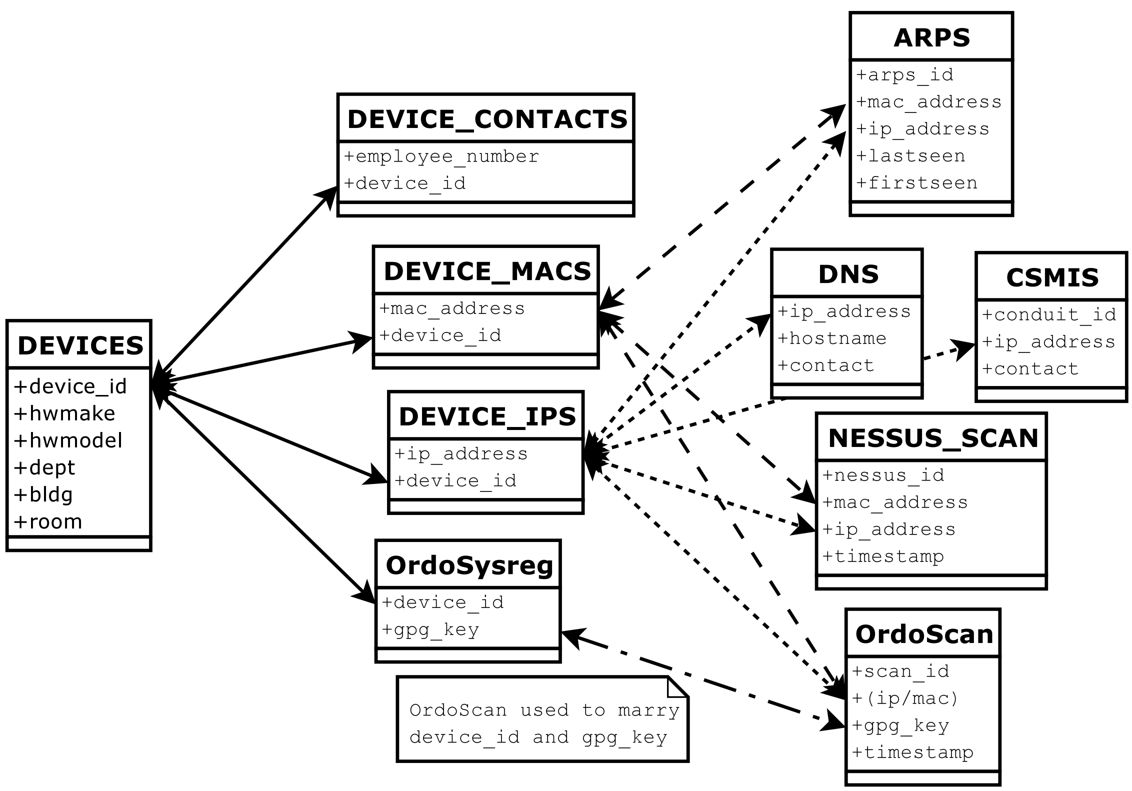

Figure 1: Simplified version of some of the top level database tables.

Figure 1 shows the simplified version of the top level database tables. In actual implementation more features than are shown here exist. In particular groups of contacts are supported as is the concept of system roles or categories.

2.3.1 Ordo Tables

The database tables related to the Ordo scanner are shown in figure 2. Details on how Ordo works are in section 5.

Each type of Ordo test (eg, NIC or FieldFile configuration file) has a coresponding table layout. The only requirement on Result table entries is that they provide a `result_id` key which points to an associated entry in the ResultInfo table. Through this, each concrete Result entry is tied via the `scan_id` entry. Finally, the full Scan is tied to a unique identification of each system, its GnuPG key id.

Other tables, not shown, hold security ratings of scan findings and remediation status.

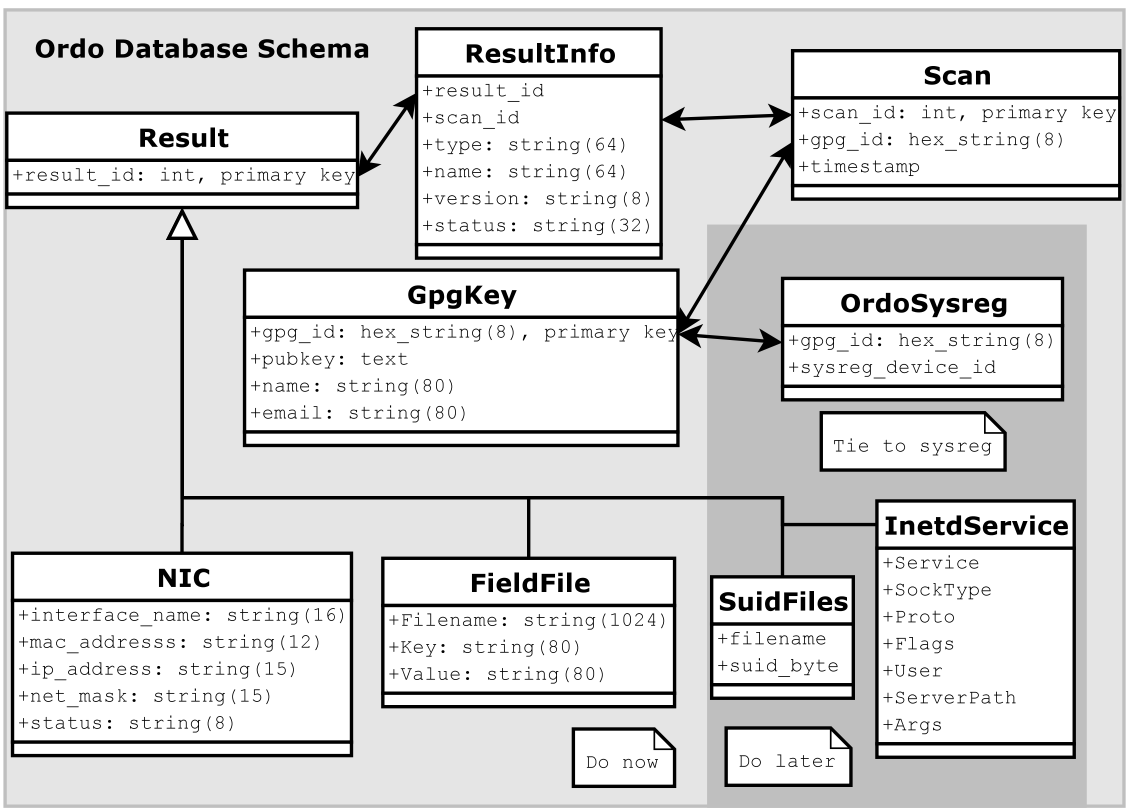

Figure 2: Basic table schema for the database back-end supporting the Ordo system.

3 Device Registration

3.1 Overview

Device registration provides a connection between network visible data (primarily MAC number) and physical data (associated individuals, location) of a system. It also is a mechanism to assure the devices that access are known in a central database. Interaction with the registration process occurs largely when a system is first introduced to the network but may be modified if network interfaces or other defining characteristics change.

3.2 Request of a Static IP

If a device has a network interface card that will use a static IP address, such must first be requested[2]. As part of that request, initial physical data is collected. This data includes the responsible individuals, location, network jack number and desired hostname. This information is recorded in the DNS database table. When a MAC is first seen using this new IP it is automatically registered with this information. If no MAC is seen within 30 days the potential registration information is discarded and a subsequent manual registration is required.

3.3 Request of a Dynamic IP

If a system requests a Dynamic IP address the DHCP server first checks if the MAC is registered. If so, it is given an IP address with full network access.

3.4 IP Use by an Unregistered System

If an unregistered system attempts to use the network either via DHCP or a hard-coded IP address which is either unallocated or allocated to another system its network access is curtailed.

4 Network Vulnerability Scanner

4.1 Overview

To locate potential network accessible vulnerabilities a scanning and remediation system based on Nessus has been developed. Nessus is a modular framework for writing network based vulnerability and penetration testing. The BNL system adds to this a database back-end storage, a web interface for initiating scans and for remediating any findings. Each module tests for a specific, known vulnerability and is categorized as low, medium or high risk. In addition, BNL selects certain potential vulnerabilities as "BNL high" risks to emphasize their importance.

Systems are scanned with different schedules based on their roles.

- All systems are scanned continuously with a select few (30) modules that check for particularly important vulnerabilities
- All systems are scanned quarterly with the entire Nessus suite of modules (1000)
- Systems with conduits are scanned several times a week with the full suite and are checked for open ports daily.

[2] http://info.itd.bnl.gov/ipreg/

5 Host Vulnerability and Baseline Configuration Scanner

5.1 Overview

For Unix-like operating systems[3] a scanner has been developed[4] to locate potential host based vulnerabilities and assess a baseline configuration requirement is met. This baseline is derived from the CIS benchmarks[5]. The complete package is dubbed *Ordo* and is comprised of the following major subsystems:

Client: A data collector running on each host

Database: Database schema and back-end

Master: Middleware to collect and update the database from client data

RWI: Remediation web interface.

DOS: Hooks into the Denial Of Service system

5.2 Client Data Collector

The client runs in a normal, but dedicated user account named *ordo*. This account lacks root privileges. Any additional privileges required to collect data are handled by group permissions.

Installation involves downloading a tar file from a known internal server. Unpacking it in a system-local directory and running an initialization script. This script performs the following tasks:

1. Determines the location of certain critical executables (perl, gpg) and the installed Ordo client software.
2. Requests identifying contact information for the client (name and email).
3. Generates a main script for initiating subsequent client actions.
4. Generates a unique client public/secret GnuPG key pair.
5. Downloads, imports and trusts the Ordo Master GnuPG public key.
6. Encrypts the client public key with the Master's public key and registers it to the Master via an SMTP message.
7. Adds crontab entries for later running of scans.

The client is then called periodically from *cron*. A run goes through these stages:

Update the client code if necessary.[6]

Scan the system using the set of tests corresponding to the given scan type.

Report the results if necessary.

The update phase first access the latest client software release file via HTTP. The release file consists of a version string followed by the MD5 checksums of all the source files. It is encrypted (signed) using the Master's secret key. If the version is newer than the currently installed file, the client downloads the corresponding tar file and unpacks it over the old version.[7].

The periodicity of scanning depends on what tests are to be run. This allows low impact tests to be run more frequently than CPU or I/O intensive ones. During installation the times in the crontab entries are chosen randomly to avoid spikes in the message rate to the master.

[3] A large majority of devices running the MS Windows operating systems are centrally manged with SMS and not considered in this section.

[4] At time of writing some aspects have not been completed.

[5] http://www.cisecurity.org/

[6] Fixme: currently not done automatically.

[7] Fixme: currently the client does not check the validity of the tar or individual source files.

The scan itself is structured by first running several data collection tests followed by a filtering or data reduction step. The result is a Perl hash following a few simple format conventions. It is organized as a Perl associative array (hash), indexed by a unique name identifying a particular test which produced the result. Each result has a consistent set of information in addition to the resulting data.

- Type of test (determines format of resulting data and corresponding database table name)
- Name of result, unique to the scan, reflecting any possible parameterization of a test.
- Version of test's code
- Unix Timestamp of when the test was run
- Result data in a well defined format specific to the test type.

The format of the result data is dependent on the needs of each individual test. This is beneficial for writing a wide variety of tests but is also means that the Master software must have intimate understanding of what each individual test supplies. In general most result data is an array of array or a key/value hash.

The full result is finally turned to a text representation of the data using the Perl DataDumper module and compared to the previously existing result, if any. If they differ or if the last scan was older than some duration, the results are sent to the Ordo Master.

All results are encrypted (signed) by the client secret key and encrypted with the Master's public key before being sent via SMTP. This assures two things. The data is secure from any potential sniffing and is resistant against falsification by third parties. It is still possible to send fake data, but any attempt at slandering a system can be proved false as long as the client secret key is not compromised.

5.2.1 Client Software Design

The client is organized as a number of Perl modules, one for each test. The modules are free to be implemented in any manner but must provide a couple of functions. The first, `revstr()` must trivially return the CVS Revision tag to provide a version string.

The bulk of the test is implemented in the second, `run()`. This function may take module specific arguments and returns a tuple containing a standard return value indicating correct operation or type of failure and the result data itself. If the test succeeds, the actual result data must be in a static format independent of arguments. If the test does not succeed any result data is ignored.

5.3 Database

The schema used for the ordo scan is shown in figure 2 and described in section 2.3.

5.4 Master

The Ordo Master is comprised of a pair of programs running on the Ordo server machine. The two programs are a mail filter, and database updater. They run on a non-privileged user account and are responsible for receiving data from the clients and updating the database on the backend, respectively.

Communication from the Clients to the Master are done over SMTP and are encrypted using GnuPG to ensure confidential transmission of possibly sensitive data (passwords, configuration data, etc.).

Communication from the Master to the Database is done over a local connection, so no Ethernet traffic is generated, also reducing the risk of sensitive data being seen on the network.

All communications from the Clients are sent to a globally known email address that is aliased to the "master" account. Email messages are received and "forwarded" via pipe to the mail filter, which decrypts the data, saves configuration information in files, which are later read by the Database Updater and submitted to the database for later analysis and archive.

Communication is done in this way for a number of reasons.

- The use of a mail filter takes advantage of the native email mechanism for queuing and reading messages. If email cannot be received for any reason, it is queued up and delivered later.

- Many simultaneous connections are possible without creating any race conditions.
- Having the mail filter parse the messages and prepare the SQL insert statements, and save them in files, frees up the Database Updater to do nothing but periodically make a single connection to the database, read all the current files, execute the insert statements, and disconnect from the database. This reduces the number of simultaneous connections to the database down to one, allowing the database server to be optimized for throughput.

5.4.1 Mail Filter

The Mail Filter receives a message, which is parsed when it is received via the Perl module Mail::Audit. The message body is decrypted using the Ordo Master secret GnuPG key and passphrase.

The message body is parsed and compared against the server's keychain to retrieve the client key ID. If the Master fails to find the client key in the keychain an email message is sent back to the client notifying the system administrator.

The Subject of the email message determines the next action the Master takes. These are:

REGISTER The message from the client contains the client's public key. It is a request to the Master to store the key in the Master's keychain for future use and client identification.

RESULT The message from the client contains results from the client's latest scan. The following steps are taken:

- Results are decrypted and evaluated using Perl's Safe module to ensure evaluation of the data
- The result data is parsed and the query strings for insertion into the database are constructed
- All query strings returned in the email message are stored in a file, given the name of the client's key ID with a timestamp concatenated, so the files for a given client can be read in the order they are received when being collected for insertion into the database

UNREGISTER The message is telling the Master to remove the client's key ID from the Master's keychain.

REREGISTER The message is telling the Master to remove the client's current key ID from the Master's keychain and replace it with the key contained in the message.

All unknown message types are forward to the Ordo administrators for analysis.

5.4.2 Database Updater

The Database Updater is a simple program that runs constantly. It behaves as follows:

1. Wake up periodically (5 minutes) from self-induced sleep
2. Try to make a connection to the database
3. If a connection is made:
   (a) Get the names of all insertion files written since last period
   (b) Read all files, storing data in hash, sorted chronologically by key ID
   (c) Perform database insertions until data is exhausted
4. Go back to sleep

5.5 Remediation web interface

To be written.

5.6 Denial of Service Hooks

To be written.

5.7 Trust Issues

There are various issues of trust in Ordo. They are listed in no particular order.

Mail message content is valid Perl data. To turn the mail message into Perl data structures the DBU must trust that the message content does not foil Perl's Safe modules protections.

Scan data is not falsified. The data can be falsified in two ways. First, the system administrator is implicitly trusted not to intentionally falsify scan results. Second, the system can be compromised and the falsified data injected by the attacker. When a compromise is detected all existing scan data from the host is not trusted until the compromise is removed.

Scan data is attributed to the correct host. Since results are delivered via SMTP, bogus data could be sent by any individual able to access the SMTP server (bnl.gov). By forging SMTP headers and learning MAC/IP addresses through ARP this bogus data could be attributable to other hosts. Because of this no data is accepted unless it is authenticated. The nature of this authentication is explained below.

Scan data is not divulged to unauthorized individuals. It is trusted that the scan result data is not divulged to unauthorized individuals once it leaves the host. Ordo encrypts the data for transmission over SMTP to the DBU process using a public key. The DBU process decrypts it with its private key and loads the data into the database. Details on this encryption are below. Once there, the database's access control lists prevent unauthorized access to the data. Front ends must authenticate and authorize any access before presenting any result data.

5.8 Result authentication and encryption

As stated above, result data is not accepted unless it:

- includes a matching fingerprint of the public key
- is digitally signed with the host's private key
- is encrypted with the server's public key

If the *ordo* account on a host is compromised any new data signed by the key or past data associated with the fingerprint is considered forever untrusted.

The fingerprint is used to uniquely identify the host and to provide an identifying characteristic that remains constant when others like MAC address change.